Analyzing Clickstreams

Andersen, Jesper; Giversen, Anders; Jensen, Allan H.; Larsen, Rune S; Pedersen, Torben Bach; Skyt, Janne

Publication date:
2000

Document Version
Publisher's PDF, also known as Version of record

Analyzing Clickstreams Using Subsessions

Authors: Jesper Andersen
Anders Giversen
Allan H. Jensen
Rune S. Larsen
Torben Bach Pedersen
Janne Skyt

Technical Report 00-5001

Department of Computer Science
Aalborg University

Created on October 31, 2000

Analyzing Clickstreams Using Subsessions

Jesper Andersen† Anders Giversen† Allan H. Jensen*
Rune S. Larsen* Torben Bach Pedersen‡ Janne Skyt†

† Department of Computer Science, Aalborg University, Denmark,
email: {spawn,giversen,skyt}@cs.auc.dk

‡ analyze.dk, Århus, Denmark, email: analyze@analyze.dk

* Nykredit, Aalborg, Denmark, email: {ahj,rula}@nykredit.dk

Abstract

Analyzing data obtained from web server logs, so-called "clickstreams", is rapidly becoming one of the most important activities for companies in any sector as most businesses become e-businesses. Clickstream analysis can reveal usage patterns on the company's web site and give a highly improved understanding of customer behavior. This understanding can then be utilized for improving customer satisfaction with the web site and the company in general, yielding a huge business advantage.

In this paper, we present the results of a clickstream analysis project at a large Danish mortgage provider. The paper first describes clickstream data and its usefulness, then it introduces the questions that the company wanted answered in the project. One of the major problems in clickstream analysis is *sequences* of clicks, which are difficult to handle using normal techniques. This problem is handled by introducing the concept of *subsessions*, which captures sequences of clicks explicitly. Techniques for overcoming the potential explosion in the number of subsessions and for filtering out unnecessary web requests are presented and the effectiveness of the techniques is evaluated. The proposed approach has been successfully implemented and tested and is currently being integrated in the company's web system architecture.

1 Introduction

The opportunities in analyzing web log data are quickly dawning on companies with web sites, as the process of improving web sites can benefit tremendously from using information on how the web sites are used. One exciting way to improve a web site is by introducing personalization to suit the different needs of different users. The sites employing personalization will give a better user experience, because users use fewer clicks to get what they want or because the content of the site is more relevant to them. For example, news web sites will show you the news that has your interest, and e-commerce sites will show you the products that you are most likely to buy. This benefits the merchant by enhancing sales and the customers by allowing them to spend less time getting what they want. The most advanced personalization sites do not require the user to perform the personalization manually by selecting desired categories of information, instead it performs the personalization based on the users *actions*. These sites are often referred to as *adaptive sites* [11, 12].

However, to be able to perform the personalization automatically, the web system has to analyze the user's previous behavior carefully to determine their habits and preferences. Thus, it is very important to find powerful and effective ways to analyze large amounts of clickstream data, especially ways of analyzing *sequences* of clicks. Current techniques have inherent limitations in terms of what information can be derived easily and efficiently, especially when analyzing sequences of clicks. The focus of this paper is to explore these limitations and come up with improved methods. Thus, using a categorization of e-business queries [13], we focus on "website design & navigation analysis."

We base our work on the current industrial state-of-the-art for data warehouse clickstream modeling [9]. Kimball proposes two star schemas, one with a click fact table and the other with a session fact table. We expand this significantly with the introduction of a new way of modeling clickstream data—the subsession fact table. By handling sequences of clicks explicitly it enables us to perform new kinds of analyses easily and efficiently.

The results described in this paper come from a clickstream analysis project for a large Danish mortgage provider, Nykredit [10]. The project was carried out as a cooperation between Nykredit, the Department of Computer Science at Aalborg University, and the OLAP consulting firm `analyze.dk`. The developed analysis system is currently being integrated in the company's web site architecture.

It is shown how two specific analysis tasks are solved, but the presented solution is general enough to handle a great number of analyses related to sequences. The company's wish is to identify places in their web site where users leave unintended, so-called "session kills". These places can be in the form of a single bad-written page, or a wrong site structure, where a sequence of pages has the undesired effect. The latter problem cannot be answered neither practically nor efficiently using the current fact table designs [9]. It is demonstrated how the subsession model is superior in this area by exemplifying how the problem of identifying sequences of pages, which lead to session kills, is solved. Another wish of the company is to identify front page banners pointing deeper within their web site, and measure their effectiveness. It will also be demonstrated how our design is superior to the existing ones for this purpose.

A great deal of previous work has dealt with the analysis of clickstream data. A large number of specialized web log analyzers have appeared [1], e.g., SpeedTracer [16] from IBM. The tools generally work directly on the raw log files without building a "real" Data Warehouse (DW) [4], meaning that they can only perform a set of pre-defined analyses, rendering them quite inflexible compared to real On-Line Analytical Processing (OLAP) systems. Another line of research [17, 6, 5] propose to build a DW with clicks as the facts, this allows OLAP techniques to be used for simple single-click-related analyses. However, analyses of sequences have to be performed using specialized complex data mining techniques, which are tailored for one specific analysis; a quite inflexible solution. Also, the complex data mining techniques cannot be efficiently supported by performance-enhancing OLAP techniques such as pre-aggregation [14]. Ralph Kimball's latest book [9] is the current "industrial standard" for clickstream modeling. Here, a DW contains only click and session facts, which means that questions about sequences are hard to answer compared to our approach. Work has been done on combining clickstream data with other customer data to get a more complete picture of the customers [2]. However, this work considers only complete sessions, not short click sequences.

We believe this paper to be the first to consider the analysis of *sequences* of clicks using standard SQL queries over star schemas [7]. Using the standard techniques enables the use of pre-aggregation, providing much faster data analysis than with other approaches, e.g., data mining. The proposed design can achieve fast query performance using pre-aggregation with any OLAP system, whether it is relational OLAP [15] or multidimensional OLAP [14]. The techniques proposed for avoiding an explosion in the number of subsessions, and the practical evaluation of their effectiveness, we also believe to be novel.

The remainder of the paper is structured as follows: Section 2 offers an overview of clickstream data and its uses. Section 3 introduces the case study, including the analysis requirements of the company. In Section 4, the case is used to illustrate the weaknesses of existing clickstream models. Section 5 introduces the new subsession star schema and use it to model the case. In Section 6, we show the usefulness of the model by exemplifying how the problems of the case study are solved using subsessions. Finally, Section 7 concludes and offers directions for further research.

2 Clickstreams

As the web grows and organizations move some or all of their business to the web, the opportunity for logging and analyzing user behavior grows. In this section we explore the types of information that clickstream data contains, along with its uses. First, we examine what clickstream information can be used for. Then we explore what data these uses are based on. Some of these uses raises privacy concerns, which we look at in Section 2.4.

2.1 Clickstream Information and Its Uses

Clickstream information can be derived from "raw" clickstream data. Here we discuss some of the most important uses inspired by the needs of the company as well as external literature [9, 3].

An important use of clickstream information is *personalization* of the site to the individual user's needs, e.g., delivering services and advertisements based on user interest, and thereby improving the quality of user interaction and leading to higher customer loyalty. It is important to *identify user groups* and focus the site towards them. Not all users are equal, some are more profitable to the organization than others, e.g., because they are more likely to buy high-profit products. Thus, it is necessary to make sure that the site caters to the wanted users. Using the groups, it is desirable to *analyze banner efficiency* for each user group, and to optimize this by presenting the right banners to the right people. *Analysis of customer interests* for product development purposes is also very interesting. Customer interests are based on what a user have looked at on the site and especially which sections of the site a user visits and spends some time in.

One of the design goals of most sites is to have a *high stickiness*, meaning that users spend a long time productively on the site. A way to achieve this is by identifying pages that often lead to undesired termination of user sessions, so-called "killer pages." It is important to find out if and why a page is killing sessions for certain user groups, and popular for other groups. Another interesting questions is whether the web site is used for different things on weekdays, weekends and holidays and if banners are used differently by the same users at different times. Finally, cost/benefit analysis of the site, or parts of it, is of great value in order to determine if it pays off, i.e., if the number of users using it justifies the current cost of updating the site?

2.2 Clickstream data

The data needed to fulfill the above-mentioned tasks is derived from a web server log file, maybe supported with cookies which are explained in the next section. From a web server log file in Common Log Format (CLF), for each click on the web site we know the *IP-address* of the user, the *page* (URL) the user requested, and the *timestamp* (down to seconds) of the event.

From these 3 facts, we can derive a great deal of extra information. First, we can identify *individual users*, through the IP-address, and through the IP-address the users *country of origin*. We can identify *user sessions*, including start page, end page and all pages visited between those two. A *user session* is an ordered sequence of clicks made by the same user, starting from the click where the user enters the site until the last click registered in the log for that user during that visit. The exit point, i.e. end page, is determined from a maximum time-span "allowed" between clicks. This prevents that the entrance click in the same user's next session is registered as part of the previous one. Also, it is easily derived what *time of day* and *day of week* a user is on the site, *how long* a user is on each page (except the last page of a session,) the number of hits per page per user, and the number of hits in a specific site section per user. Finally, we can *group users* by time between clicks and the time of day/week users are on the site and count the number of hits on a banner, totally or by day/week.

2.3 Cookies

Using clickstream data alone, it is not possible to recognize users with dynamic IP-addresses, which change from session to session. Consequently, private users connecting through an Internet Service Provider(ISP) cannot be identified and distinction of users behind a proxy server is not possible. By placing a cookie on a users computer, it is possible to trace a user with a dynamic IP-address through more than one visit. Using cookies we also know if a returning user has a static or dynamic IP-address and if the user is behind a multi-user proxy-server. This is indicated when different cookies are found from the same static IP-address. We can also see cookies from other web sites and use them to create a more detailed user profile. It is possible to read cookies from other sites by exploiting a security weakness in the current version of Internet Explorer. The privacy concerns this raises are, amongst others, discussed in the next section.

2.4 Privacy Concerns

As the importance of the web grows with users extended use of the internet for shopping, etc., a big issue arises; namely privacy—how much can businesses learn about you by logging your actions on their web site?

By using a weakness in the current version of Internet Explorer it is possible to get cookies from other web sites a user has visited. The means to do this is redirecting a user to a special URL, which sends a specific cookie to the server. You have to know what you are looking for, i.e., to get a cookie you have to specify from which site you want the cookie. If the user's name is present in a known cookie, the security hole can be used to get this information.

Adding information from the other sites a user has visited enables extremely precise profiling of the user, which among other things can be used to emphasize the areas, where your company does better than the other companies, that the user has visited, in the same area of business. By combining clickstream data with "borrowed" cookies from other web sites a more personal site can be made. The possibility for abusing the "open cookie jar" is clear. For instance it is possible to get a *hotmail.com* cookie from a user, and then read the user's mail if he/she has got a hotmail account.

If it is desired to be anonymous while browsing the Internet, it can be done. Of course, by being complete anonymous the benefits of personalized services are lost. To be anonymous, cookies must be disabled, and the users need to make sure that their IP cannot identify them. If IPs are assigned dynamically from your Internet Service Provider (ISP), the user can simply hide among the others that use the same ISP. The only thing others can tell is the ISP and country. The same applies if users are behind a multi-user proxy server. If users are in the position of having a fixed IP, it is possible to buy a service on the web which hides your IP-address.

Special privacy concerns are present when dealing with e-commerce. Contrary to normal shopping, users cannot be anonymous because they have to give their addresses to the seller so the seller knows where to send the purchased goods. Moreover, the users have to give the seller credit card information in order to pay for the goods. The credit card information is the seller's security that you will pay. Your security is based on trust in the seller. Thus, the seller must have a good reputation.

The personal information the seller has about the users can be misused. For instance, the seller can charge more money than expected on your credit card. The businesses that abuse the trust of their customers will be unsuccessful—rumors spread very fast on the web. But those that use the information to deliver a good personalized and trustful service will be very successful. Building a trusted name in their customer group is crucial for e-commerce companies. The means to do this is general good service and the word will spread. The other possibility is advertising, using the Internet or other channels such as TV and radio.

3 Nykredit Case

The case of clickstream analysis in this paper involves the large Danish mortgage provider, Nykredit. This company has a web site [10], from which five months of clickstream data was obtained. Nykredit had a number of questions that they would like to answer using clickstream analysis. Among these, this paper focuses on the following two challenging problems.

The site has *banners* for advertised products on the front page, they direct users to deeper lying sections of the site. To maintain user interest these banners are changed regularly. The task is to identify *which* banners have been up, *when* they were up, and *how effective* they were.

A wrong sequence of pages can lead to users loosing interest in the site and *leaving* at unintended times, e.g., if users cannot find what they are looking for or suddenly find themselves in an unwanted part of the site. We wish to isolate these sequences of pages, so-called *killer subsessions*.

To solve these problems efficiently, we model our clickstream data in a data warehouse, and design queries to extract the wanted information. The existing modeling approaches [9] are not optimal for solving the problems in our case, as will be shown in Section 4. This is the motivation for introducing the new modeling approach in Section 5, based on modeling subsession.

4 Existing Clickstream Models

The most commonly recommended clickstream models are *click fact models* and *session fact models* [9]. However, as we argue, the analysis in focus in the Nykredit project either cannot be done in the context of these models, or they

will perform very badly. The project has the two queries mentioned in Section 3 in focus. However, these are only examples of many similar queries based on clickstream data.

Before discussing the capabilities of the two models in relation to our specific purposes, we briefly introduce a simplified version of each of the models presented in [9]. A more detailed description can be found in Appendix A.

Click Fact Modeling A click fact model is a star schema modeling *single clicks* in the fact table. It has the dimensions *date*, *time_of_day*, *session* which captures the session that the click belongs to, and *URL*, which captures the requested web page. The fact table itself will typically model information like *number_in_session* and *click_seconds*, where number_in_session illustrates where in the user session this specific click is placed.

Session Fact Modeling Session fact models are a recommended alternative and supplement to click fact models. Instead of single clicks, it models *full sessions* of clicks in the fact table, again using the dimensions *date*, *time_of_day*, *URL*, and *User*, which captures the IP address of the user. The fact table will typically model start_page and end_page using the URL dimension, but information on the internal clicks in the session will be lost.

One advantage of using the click fact table is that data is loaded into the warehouse at the same level of detail as found in the web log. It has the same fine granularity and no detailed information is lost in aggregations. The cost of this is that the connection of the clicks, in terms of sessions or subsessions, is only accessible indirectly, and this is the main problem for the queries in question in the Nykredit project (see Section 3) as both require finding click sequences of length two or more.

The problem is illustrated in the query presented in Appendix B. The query shows how multiple self-joins on the fact table is necessary to select even short sequences of successive clicks from the same session. Furthermore, unions of such multiple self-joins are needed to work with sequences of different lengths, leading to a very complicated query. There are several problems associated with this query. First, a query this complicated is very error-prone in terms of formulating the query. Secondly, the consequence of multiple self-joins on a large fact table is a bad query performance, see Appendix B for details.

Turning to the session fact table, the advantage is that it is easier to follow a user through the site. However, since no internal clicks are modeled, the detail knowledge available in the data suffers fatally. This is because it is impossible to pick out a single click. The result is that full clickstreams are directly accessible, but it is impossible to compare and see if two full sessions start or end in the same shorter sequence, i.e., to see session kills or the effectiveness of front-page banners.

Thus, neither the click fact schema nor the session fact schema solves the problems in the Nykredit project. Both are described as common ways of clickstream modeling [9], but the former requires evaluation of complicated queries and consequently has bad performance, while the latter does not preserve the relevant data from the web log. Thus, they are not usable for the purpose described in Section 3.

5 Subsession Fact Modeling

To solve the problems set up in the Nykredit project, we now introduce a new model based on subsession facts, since the analysis needs focus on parts of sessions (see Section 3).

5.1 Subsession Fact Model

The idea of the subsession fact model is to model every possible sequence of successive clicks from each session. We call such a sequence a *subsession*. An average user session will span many subsessions, and the subsessions will overlap each other many times. This is exemplified in Figure 1 where a 4-click session is cut into 6 subsessions. Subsessions of length 1 are not included since they are just single clicks and therefore better modeled using a click fact model.

The subsession star schema is seen in Figure 2. The *Time Of Day*, *Date* and *Session* dimensions are all very similar

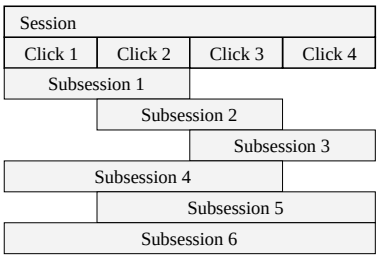

Figure 1: Subsessions Spanning a 4-click Session

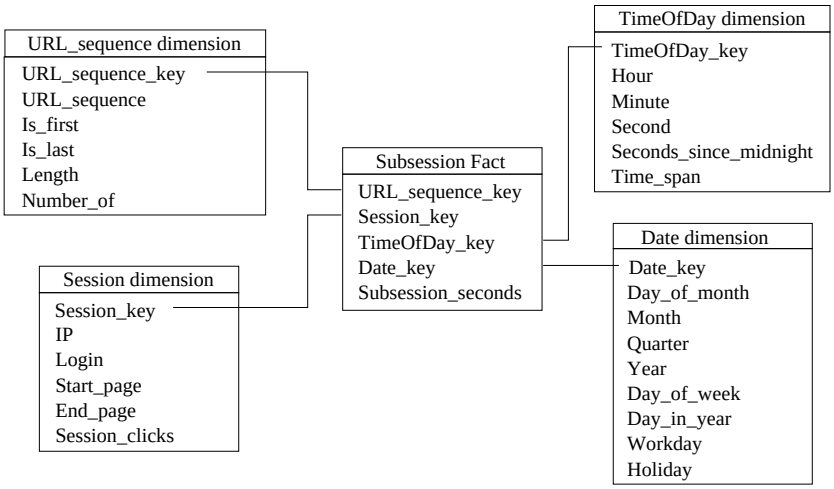

Figure 2: Subsession Star Schema

in structure to the proposed models in [9]. The *subsession* fact table contains one piece of aggregated information: *Subsession_seconds* measuring the time-span of the subsession in seconds. This was chosen with the assumption that the information could be useful in improving particular subsessions, where users spent inappropriately long time or as part of user profiling. These queries are not directly part of the Nykredit project, but are very interesting for future developments.

Let us for a moment look at the parallel to voyage databases. A voyage database is a database of the traveling done by airline passengers between cities [8]. Our passengers are web site users, and the airports are pages on a web site, but the problem is the same, we need to model traveling. The application of voyage theory to clickstream analysis is possible when analyzing how users travel in the network of web pages. Modeling this network is a key element in modeling the usage information from the clickstream.

We store the network information in the *URL_sequence* dimension. This dimension has the attribute *URL_sequence* which represents the sequence of URLs that the referring subsession consists of, i.e., all the URLs combined in the order they were visited. Thus, it is not possible to efficiently access single URLs individually for querying, instead it is possible to make very fast queries on entire subsessions.

To speed up queries, we add derived data to the model. In [8] Ralph Kimball exemplifies this problem on a Frequent Flyer Data Mart schema where Trip Origin and Trip Destination are added, using the argumentation: *"This design is almost useless until the two extra dimensions are added. The design then becomes very powerful, and can answer most marketing questions."* The marketing questions from clickstream analysis are different from those of airlines but both can be considered voyages in networks. If you want to know why the user are doing what they do, the first question is where they are going.

The URL_sequence dimension contains four aggregates: *Length* measures how many clicks are represented in this subsession, *Number_of* counts the number of fact table entries that are instances of this subsession, and finally *Is_first* and *Is_last* count the number of instances of this subsession that start or end the session it comes from, respectively. The latter three are counted from the fact table entries that have the URL_sequence in question.

We have chosen to include the latter three aggregates in the dimension table to be able to explicitly stress the advantages in terms of of performance of having such pre-computed aggregates. However, in a "pure" design, such aggregates should be implemented using materialized aggregate views, making their existence transparent to the users of the DW. The aggregated information is very useful for the analysis made in Section 6, where we extract information regarding killer subsessions and front-page banners.

In general, the subsession star schema provides the data mart to solve the specific problems in our case and similar problems. However, other analyses might benefit from having additional dimensions and/or aggregates.

In the context of the Nykredit project, the subsession model is superior to any other known model as it is the only one that supports efficiently answering questions about sequences of clicks with click fact precision. The query performance of the subsession fact model is superior to that of the click fact model since it is not necessary to access each click individually when information about a subsequence of a session is desired. For many of these queries we do not even have to access each subsession, as the aggregates in the URL_sequence dimension encompass all instances of subsessions that have followed that URL sequence. An example of this is counting the number of times a URL sequence has been the last in a session.

However, a potential problem is the model's space efficiency which is explained in the following section.

5.2 Breeding Subsessions from Sessions

A downside of subsession modeling is the large use of secondary storage. The extent of this problem is discussed here, along with possible ways to minimize this problem.

A session can be modeled with a fact entry for each way of dividing it into a subsession. For example, a 20-click session would give one 20-click subsession, two 19-click, three 18-click and so on. A total of 190 subsessions excluding trivial one-click subsessions[1]. This maximum number of subsessions can be formulated as *maxss*, which is a function of the session length, sl.

$$maxss(sl) = \sum_{i=1}^{sl}(i-1) \;\approx\; 0.5 \times (sl-1)^2 \tag{1}$$

The simple definition in (1) shows a squared dependency of the session length. A 1-click session has no subsessions, and for each single click added to the session length, i.e., now $(sl+1)$, this session length minus one, sl, is added to the maximum number of subsessions. This quickly becomes a large number as seen in Figure 3, and will

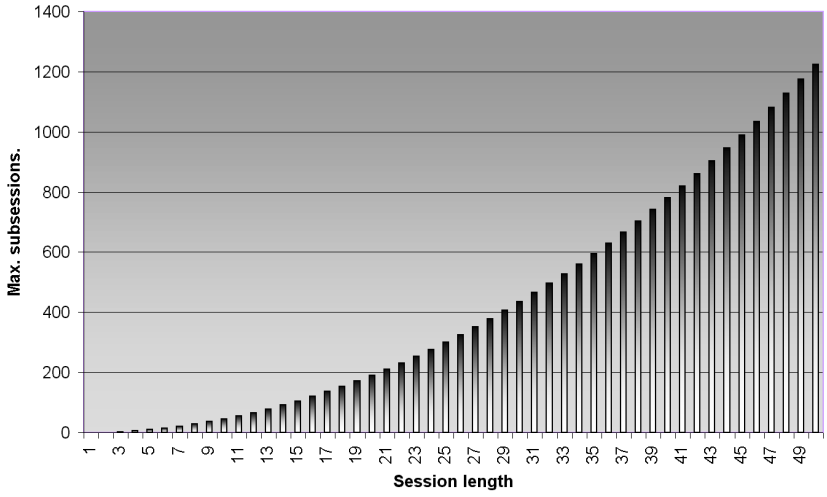

Figure 3: Maximal Number of Subsessions

[1]One-click subsessions, i.e. clicks, are better modeled using a click fact table.

cause the fact table to grow rapidly as longer subsessions enter. As an example, for each session of length 50 there will in average be 24.5 subsessions starting per click.

We have extracted the distribution of session lengths from the clickstreams in the case study, see Figure 4. Combining this with (1) gives us a precise prediction of the average number of subsessions we will have starting

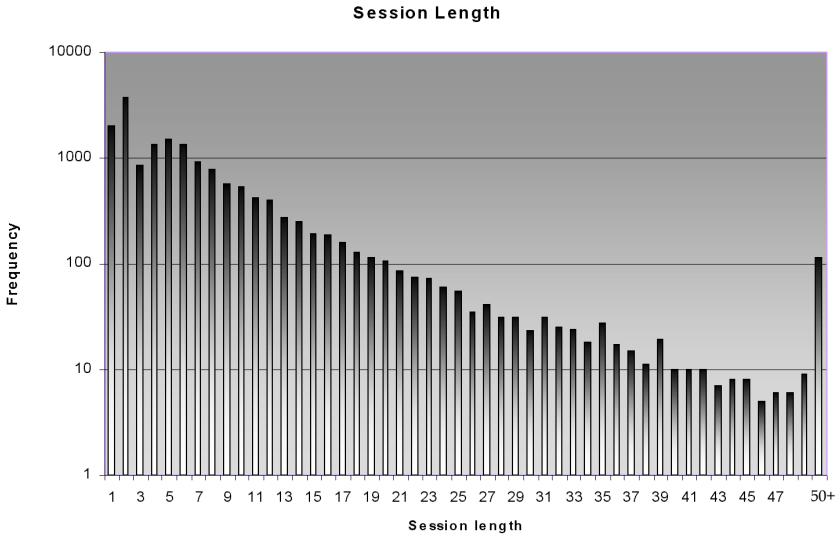

Figure 4: Session Length

from each click in the clickstream, and thus how large the requirements for secondary storage is relative to the click fact model (for this specific business model). In our case, the requirement is 3.56 subsessions starting per click (an average weighed against the frequence of session lengths.) If the average session is longer than in our case, the storage requirements will be even larger.

Thus, handling the space requirements of subsession modeling can be a practical problem, which sets high demands to disk space. Therefore, it is necessary to investigate techniques to reduce the number of subsessions. Three such techniques are proposed in the next section.

5.3 Choosing between Subsessions

Choosing which subsessions to include and which to exclude must be based on the organization's desires in terms of analysis and use of storage space.

If the specific tasks to solve, and the corresponding storage requirements, are already known, choosing subsessions to fit these is relatively easy. The big problem consists of satisfying the demand for generality ensuring that the modeling will be useful for yet unknown, future queries. The solution to this problem is highly dependent on the organization and its criteria for the clickstream analysis.

In the Nykredit project, excluding unusable subsessions was not a satisfying solution. All subsessions were needed because of the approach chosen for identifying session killer subsessions (see Section 6.3.)

Instead of choosing among the subsessions after relevance, reducing the subsession number can be done by limiting the length of subsessions. The memory of a user stretches back only a limited number of clicks, so setting a maximum length of subsessions will most likely not have a high impact on behavioral analysis. If analysis of whole sessions is desired, a session model should be used.

Setting a maximum subsession length is of course associated with a slight cost in data quality, i.e., analysis of long sessions. These would have to be divided into smaller pieces, and the pieces put together. However, the detail achieved from having information on the pieces may often outweigh the extra information which is only represented in the whole. It is very hard to interpret the analysis that long subsessions provide since it spans a large and sometimes diverse area of the site, thus useful and unified conclusions are difficult to draw.

A final reason why a maximum subsession length limit does not influence most clickstream analyses is that there is much less data supporting statistical analysis on longer subsessions than on shorter. The likeliness of a sequence of clicks being traversed naturally decreases when its length is increased since users have more places to abandon the given subsession before finishing the sequence of clicks. This is also demonstrated by the clickstream data from the Nykredit project.

Setting a maximum length of subsessions, $maxSSL$, gives a new formula for the number of subsessions, ss, in a session of length sl:

$$ss(sl) = \left(\sum_{i=1}^{min(sl,maxSSL)} (i-1) \right) + (sl - min(sl, maxSSL)) \times (maxSSL - 1) \quad (2)$$

Equation (2) contains the same summation as (1), except for the upper limit of $maxSSL$ for i. The product added in the end applies, when the session is longer than the allowed subsession length. In this case, the extra clicks are prohibited from causing the subsessions to increase as fast as before. From this point the growth in subsessions is linear. The reason for using $maxSSL - 1$ is that only subsessions of length 2 and up are modeled.

In Figure 5 this is exemplified with an arbitrarily chosen max subsession length of 8. This does not influence the desired analysis since none of the tasks in the case study required longer subsessions to be modeled, see Section 6.4 for further discussion. This gives us a up to 73% reduction in the number of subsessions bringing it down to 6.4 sub-

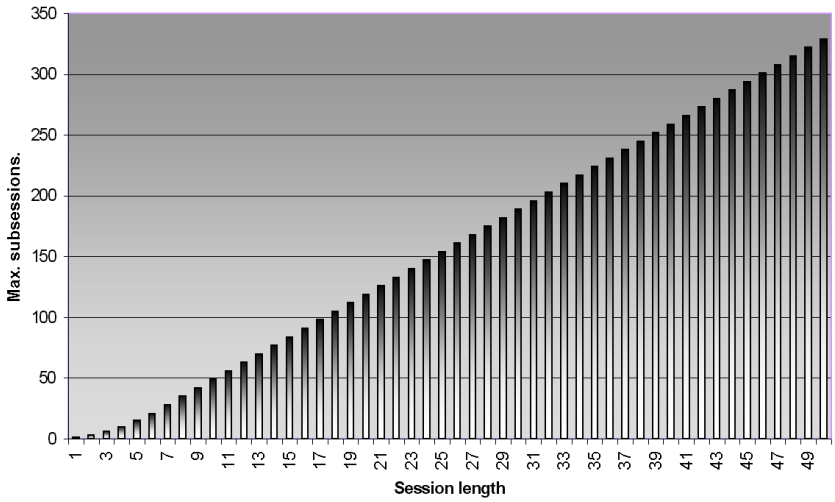

Figure 5: Number of Subsessions with $MaxSubsessionLength = 8$

sessions starting per click for sessions of length 50. Given the distribution of session lengths in the case study, the average number of subsessions starting per click is 2.78 which is a 22.2% reduction compared to keeping all subsessions. This is a more manageable size given our premises, but it could be expected even better. The explanation is the large number ($> 75\%$) of sessions of length $1 - 8$. Since they still cause the same number of subsessions in the fact table, the improvement is only based on less than 25% of the sessions. For comparison, a maximum subsession length of 5 will for the case study give a 32.9% reduction in the number of entries in the subsession fact table.

The third possible subsession reducer is starting at the other end, setting a minimum subsession length, $minSSL$. If subsessions of length n and below are irrelevant for the analysis, a significant reduction of fact table size can be made from only looking at subsessions of length $n + 1$ and up.

The formula (3) is almost the same as (1), only with the exception that the summation stops earlier, when it reaches the number of subsessions having the minimal subsession length allowed, $minSSL$. In a session of length sl, there is $sl - minSSL - 1$ subsessions of length $minSSL$. Equation (2) can be adjusted in a similar way.

$$ss(sl) = \sum_{i=1}^{sl-minSSL} (i-1) \quad (3)$$

Setting a lower limit for the subsession length has impact on analysis quality of short sequences of pages, and because of the statistically more significant analysis results that come from analyzing shorter sequences (which occurs more often), this approach should probably not be preferred for limiting disk space use. Eliminating 2-click subsessions would in the Nykredit case give a 24.6% reduction of subsessions compared to the maximum number. However we did not do so, since we need to analyze 2-click subsessions. This reasonably good improvement is achieved since all session lengths are source for a number of 2-click subsessions.

Yet another technique for reducing the amount of data is to exclude very short sessions, e.g., of length 1 and 2, from the data set, as they are likely to represent users that entered "by mistake", meaning that they are not really useful for behavioral analyses.

To summarize, we have seen that the number of subsessions and the effectiveness of the reduction techniques vary a lot depending on the session length. A quite surprising result is the fact that eliminating subsessions of lengths greater than 8 in our case only gives a 22.2% reduction compared to keeping all the subsessions.

Thus, in our case it is feasible to keep all subsessions which provides a better foundation for advanced analyses. However, this may not be true for other web sites as the distribution of session lengths varies according to the structure and purpose of the web site in question. An example where substantially shorter sessions would be expected is search engines as *altavista.com* and the like. Having shorter sessions requires less storage, making subsession modeling an even more usable approach.

On the other hand, typical e-commerce sites, where the users search, select, and buy several products will have somewhat longer sessions. Our case is somewhere in-between as the Nykredit site primarily informs about financial products, which require more than a few clicks since financial products are more complicated to describe than the majority of e-commerce products.

6 Querying and Data Processing

This section describes extracting the information from the Nykredit case clickstream data, and using the subsession model introduced in Section 5 for analyzing and interpretation over this information. The clickstream data is represented in subsessions limited to a length of minimum 2 and maximum 8 clicks, as described in Section 5.3.

In order to gain the desired information, a hypothesis on how this information is present in the clickstream data is formalized into a database query. After extracting the data it is interpreted as a source of information to help solving the problems mentioned for the Nykredit project.

6.1 Front Page Banners

The first problem was, as stated in Section 3: *"The site has banners for advertised product on the front page, they direct users to deeper lying sections of the site. To maintain user interest these banners are changed regularly. The task is to identify which banners have been up, when they were up and how effective they were."*

To find the changing front page banners and their effectiveness we set up the following hypothesis: The percentage of clicks from the front page that lead to another page increases substantially if a banner is set on the front page pointing to that page. If a banner is set up at time t, we can formulate this in mathematical terms the following way.

$$HitsFromFrontPage(page, t+1) >> HitsFromFrontPage(page, t-1) \Leftrightarrow$$
A banner is set up on the front page, at the time t, pointing to $page$

The increases in hit ratio is found in the following way. First, hits from the front page to other pages are isolated by looking at subsessions of length 2, where the start page is the front page, this is aggregated in the fact table and thereby quickly retrieved. Second, for all pages, their hits from the front page are counted for each day. If a page never has more than a certain number of hits from the front page, it is removed from the list of analyzed pages due to statistical insignificance. Moreover, a target page of a front page banner is assumed to have more than this number of hits, so this removal of data will not affect data quality. This gives all the wanted $(page, date, hits)$ 3-tuples. Third, the hits for each day is weighed by dividing with the total number of hits for that day, i.e., each $(page, day)$ is now assigned a hit ratio attribute instead of the hits attribute. Summing the hit ratios on all pages gives 1. This eliminates

the problem of, e.g., weekend dips and highs in the clickstream data, which would otherwise look as sharp changes in activity and point us towards the false conclusion of a banner change. Finally, graphs are build and changes found based on $(page, date, hitratio)$.

Now we have a 3-D graph $(pages, day, hitratio)$ illustrating for each day how the hits from the front page are placed on the other pages of the site. The SQL for retrieving the data is seen next.

```
SELECT  u.URL_sequence, s.date, count(*) / q.datehits
FROM    subsession_fact s, URL_sequence u,
        (SELECT s.date AS dhdate, count(*) AS datehits
         FROM   subsession_fact s1, URL_sequence u1
         WHERE  u1.Is_first > 0 AND s1.URL_sequence_key = u1.URL_sequence_key
                AND u1.length=2
         GROUP BY s.date) q
WHERE   u.is_first > 0  AND u.length = 2 AND s.date = q.dhdate AND
        s.URL_sequence_key = u.URL_sequence_key
GROUP BY u.URL_sequence, s.date;
```

6.2 Front Page Banner Data Processing

The $(page, date, hitratio)$-graph created from the query inSection 6.1 will now be analyzed to find the wanted banners, and their effectiveness. The graph is seen in Figure 10 (at the end of the paper.)

The *URL Sequence* axis consists of a set of two-click URL sequences. In every sequence, the first page is number 1, i.e. front page, and the next number is a code for the following page. We will translate these codes into URLs as needed. The *Day in April* axis represents which date in the selected data of April 2000 is illustrated, ranging from April 1st to April 30th, 2000. In the *hit volume* axis, a color shift corresponds to a ratio of 0,4% of the total hits from the front page on the day in question.

The hypothesis was that a banner change would lead to a sudden change in hit activity, so that is what we are looking for. The first place this can be found is at point A. Here, activity on the URL sequence "1 19" decreases abruptly from being steady between 0,8-1,2% to below 0,4% after the April 14th. Page 19 in question is $www.nykredit.dk/privat\_kerne/notitser/boligraadgivning-for.htm?segmentid=0$, which is an A-Z guide on trading real estate, also containing banners to real estate agencies. This page must have been the target of a front page banner, because it is not possible to reach it directly from the front page.

The next interesting point is point B. B states an abrupt increase for an URL sequence from less than 0,4% to between 1,6% - 2.0%. Thus, contrary to point A, B states an increase, illustrating that a banner is placed at the front page at this time. Afterwards the use of the banner fades out to 0,4% - 0,8%. It seems that the banner to page 19 (point A) was replaced with a banner to the page 3848 (point B) on the April 14th. The new banner target (from point B) is $www.nykredit.dk/privat\_kerne/rentesikring.htm?segmentid=2$. This page is dedicated to a special loan product of the company, called "rentesikring" (English: interest rate protection). As a matter of fact, this banner is still present on the Nykredit front page at the time of writing (May 21, 2000).

Section C points out the clearest result from the analysis. The URL of this popular page, numbered 2750, is $www.nykredit.dk/privat\_kerne/notitser/vaerdipapirhandel.htm?segment=pri\&type=2\&segmentid=2$. This advertises an online stock-trading product. The hit ratios for the banner to this URL has been cut out of Figure 10 and placed in Figure 6. Here we can see that before April 8th, and after the April 27th, no requests are made for the page. However, in between these dates, the hit ratio is 1.0-3.0%. So, this has clearly been a front page banner, but is now moved to a less lucrative spot on a sub-page.

To conclude we have clearly identified three front page banners using two-click subsessions. The same technique can be applied to any page and link on the site to analyze how the site is used.

6.3 Killer Subsessions

The problem of killer subsession was stated as follows. *"A wrong sequence of pages can lead to users loosing interest in the site and leaving at unintended times, e.g., if users cannot find what they are looking for or suddenly find*

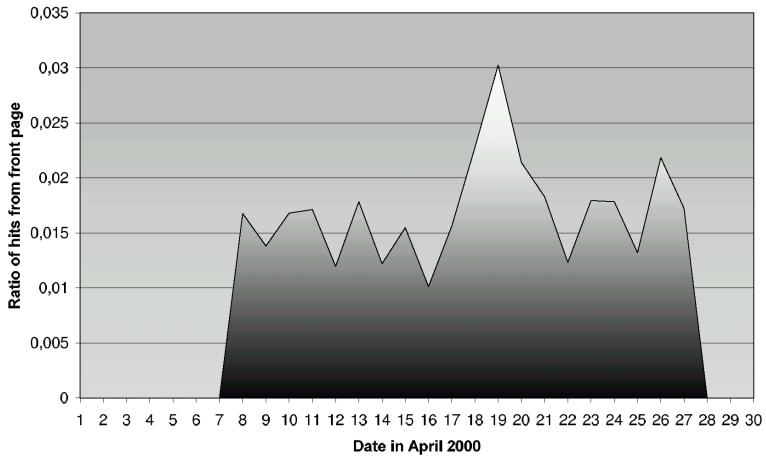

Figure 6: Front Page Banner Target Page

themselves in an unwanted part of the site. We wish to isolate these sequences of pages, so-called killer subsessions."

Again, we set up a hypothesis: If a certain subsession has a high percentage of users, who leave the site after traversing it, it is likely to be a killer subsession. In mathematical terms:

$$Sessionkills(ss)/hits(ss) > threshold \Leftrightarrow \text{The subsession } ss \text{ can be a killer subsession.}$$

The recipe used to find killer subsession is as follows. First, for all subsessions, the number of $hits$ and the number of times they led to session-kills, Is_last, are counted, Yielding the ratio between these two numbers, $killratio$, (both of these are aggregated in our model and therefore quickly retrieved). Second, subsessions with n or less hits are removed and the remaining are ranked due to their $killratio$. Setting a high n gives better statistical certainty but removes subsessions that might be interesting. Finally, graph the data and find maxima of the $killratio$.

Expressed in SQL, our data was found with this simple query:

```
SELECT   URL_sequence, is_last / number_of AS killratio
FROM     URL_sequence
WHERE    is_last > 5
ORDER BY killratio DESC;
```

6.4 Killer Subsession Data Processing

Running the query above to find killer subsessions on a month of clickstream data from Nykredit resulted in a long list of subsessions with their $length$ and $killratio$. On Figure 7, the 25 subsessions with the highest kill ratios are represented for 5 different subsession lengths. The figure illustrates that even long subsessions can be killer subsession, since it does not show a clear difference in the *kill ratio* for short and longer subsessions. However, subsessions of lengths 2, 3 and 4 are all better represented with high *kill ratios*.

Looking at specific killer subsession candidates, Table 1 shows three specific subsessions with high kill ratios. Examining the web pages visited through the subsessions, we find the candidates to be of one of the following three types.

- Pages where users have fulfilled the purpose of their visit, and intendedly leave. This does not qualify as a killer subsession.
- Pages where users are lost because of lack of clickstream data, e.g., after users have logged into the secure server from which no clickstream data is available. We are not in a position to conclude on those subsessions.
- Pages where users unintendedly leave without having fulfilled their purpose—*true* killer subsessions.

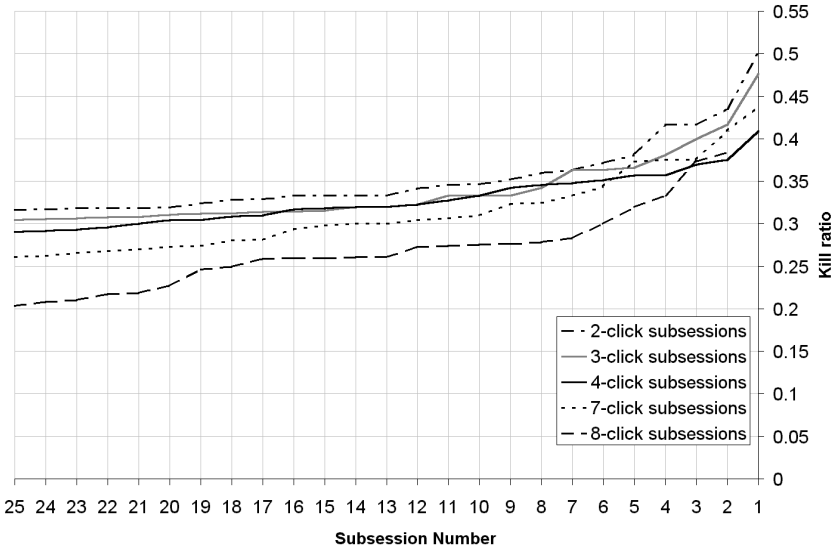

Figure 7: Killer Subsessions

URL_sequence	Kill ratio	Description
13 51 520	0.47	Page 520 consists of links to real estate agents that Nykredit cooperates with. So it is intended that users leave to one of the other sites, thus this is not a killer subsession.
41 46 10 61 62 12 41	0.44	This subsession starts and ends on the same page, which indicates that users surf in a circle causing possible frustration. However, the page in question is the answer page for loan calculations, so users leaving the sight from this place could also have fulfilled the purpose of their visit, causing the high killratio.
3678 3679	0.43	This sequence ends in a gigantic form, which users have to fill out, leading to possible frustration. This could easily be a killer subsession.

Table 1: Subsessions With High Kill Ratios

A consequence of the two "false" types of candidates is that finding killer subsessions requires interpretation to rule out the two other options. The selected subsessions in Table 1, all with high kill ratios, exemplifies the kind of interpretation needed and the possible results. We conclude that identifying killer subsessions requires substantial interpretation to come up with useful results.

7 Conclusion and Future Work

In this paper we have presented the results of a clickstream analysis project for a large Danish mortgage provider. The project has investigated the analysis of sequences of clicks using data warehousing techniques. We have introduced the subsession fact table model, and used it to solve two specific tasks for the company web site.

The first of these tasks is identification of front-page banners, including analysis of their efficiency. Using data for April 2000, the changes in front-page banners were clearly found, and the effectiveness of a banner valued. The subsession approach allows precise, fast and comprehensive banner/link efficiency analysis for all pages of the site. A strong tool for this is the advanced, usable 3-D $(page, date, hitratio)$ graphs, which can be created from the results of a single subsession query.

The other task is identification of subsessions which lead to users leaving the site prematurely, so-called "session kills." A large amount of candidate session kills were found and some were found to be killer subsessions while

other candidates had a high kill ratio for other reasons, e.g., because the users had finished their intended tasks. More interpretation is needed for this task than with banner efficiency analysis. The killer ranking of subsessions can be a tool in improving the site's "stickiness" towards users.

We have found that the main strength of using subsessions is the ability to model sequences of clicks explicitly. The main weakness of the subsession model is the potentially large use of secondary storage. This issue has been addressed, and methods set up for limiting the disk use to acceptable levels. However, in our case we found that the problem was not as bad as expected, since we could store all subsessions of any length, respectively all subsessions of lengths 2–5, using only 3.8, respectively 2.5, times the storage required for storing the individual clicks. In general, the extent of the problem varies with the average session length which is influenced by the purpose of the web site.

Overall, we conclude that the subsession approach is superior to the existing methods in answering questions about *parts* of user sessions, answering usage questions about *subparts* of a web site, including analysis of banners and links, and in providing *detailed behavioral analysis*. It is inferior to the existing click fact and session fact models in answering questions about *single pages* and *entire user sessions*.

We believe that the subsession model introduced in this paper is a significant contribution to the field of detailed clickstream analysis. Moreover, subsession modeling lays the foundation for increased web site usage analysis, user analysis and personalization possibilities.

Many uses of subsessions still lay open, and any new practical application of subsession analysis is interesting. An especially interesting area of subsession analysis which we have only touched peripherally, is the profiling and grouping of users from the behavioral data that it is derivable from subsessions. This information can be used as a basis for successfully creating or enhancing the personalization of web sites.

References

[1] A Listing of Web Log Analyzers. `www.uu.se/Software/Analyzers/Accessanalyzers.html`.

[2] A. G. Büchner and M. D. Mulvenna. Discovering Internet Marketing Intelligence through Online Analytical Web Usage Mining. *SIGMOD Record*, 27(4):54–61, 1998.

[3] R. D. Hackathorn. *Web Farming For The Data Warehouse*. Morgan Kaufmann, 1999.

[4] W. H. Inmon. *Building The Data Warehouse*. Wiley, 1996.

[5] A. Joshi and R. Krishnapuram. On Mining Web Access Logs. In *Proceedings of the 2000 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery 2000*, pp. 63–69, 2000.

[6] K. P. Joshi, A. Joshi, Y. Yesha, and R. Krishnapuram. Warehousing and Mining Web Logs. In *Proceedings of the Second ACM Workshop on Web Information and Data Management*, pp. 63–68, 1999.

[7] R. Kimball. *The Data Warehouse Toolkit*. Wiley, 1996.

[8] R. Kimball. Traveling through databases. *DBMS Magazine*, 10(5):16, 1997.

[9] R. Kimball. *The Data Webhouse Toolkit*. Wiley, 2000.

[10] Nykredit Corporation. `www.nykredit.dk`. Current as of July 12, 2000.

[11] M. Perkowitz and O. Etzioni. Adaptive Sites: Automatically Learning From User Access Patterns. In *Proceedings of the Sixth International World Wide Web Conference*, 1997.

[12] M. Perkowitz and O. Etzioni. Towards Adaptive Web Sites: Conceptual Framework and Case Study. In *Proceedings of the Eighth International World Wide Web Conference*, 1999.

[13] I-Y. Song and K. Levan-Shultz. Data Warehouse Design for E-Commerce Environments. In *Proceedings of ER Workshops*, pp. 374–387, 1999.

[14] E. Thomsen. *OLAP Solutions: Building Multidimensional Information Systems.* Wiley, 1997.

[15] R. Winter. Databases: Back in the OLAP game. *Intelligent Enterprise Magazine*, 1(4):60–64, 1998.

[16] K.-L. Wu, P. S. Yu, and A. Ballman. SpeedTracer: A Web Usage Mining and Analysis Tool. *IBM Systems Journal*, 37(1):89–105, 1998.

[17] O. R. Zaïane, M. Xin, and J. Han. Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs. In *Proceedings of the Fifth IEEE Forum on Research and Technology Advances in Digital Libraries*, pp. 19–29, 1998.

A Existing Clickstream Models

This section describes the click fact and session fact models in more detail.

A.1 Click Fact Schemas

The click fact table contains single clicks at the site, single requests for a page. In Figure 8 a click fact table is shown. It has the following dimensions. The *URL dimension* captures which URL (web page) the user has visited and the *Date dimension* captures the date the user visited the site. The *Session dimension* captures the session that this click belongs to and additional information, e.g., the start and end page of the session, while the *Time of Day dimension* captures the time of day of the request.

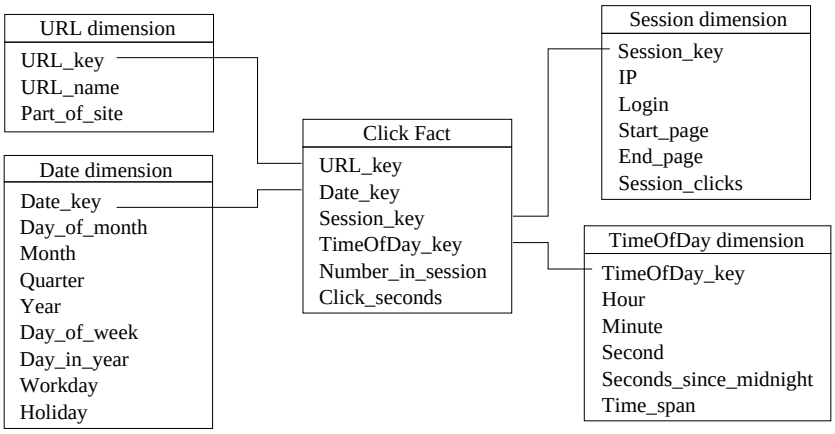

Figure 8: Click Fact Star Schema

This fact table and its dimension tables are taken from [9] and is a recommended way to create a clickstream star schema. One advantage of using click fact is the high data quality achievable because of the fine granularity, and the fact that no information is lost in aggregations. This means that we can analyze on a single-click level.

A.2 Session Fact Schemas

Session fact schemas is a model of aggregated information about complete user sessions on the site also proposed in [9]. The session fact table, shown in Figure 9, has following dimensions. The *Date dimension* captures the date the session took place. The *Time of day dimension* captures the time of day the session took place. The *User dimension* captures the user's IP address.

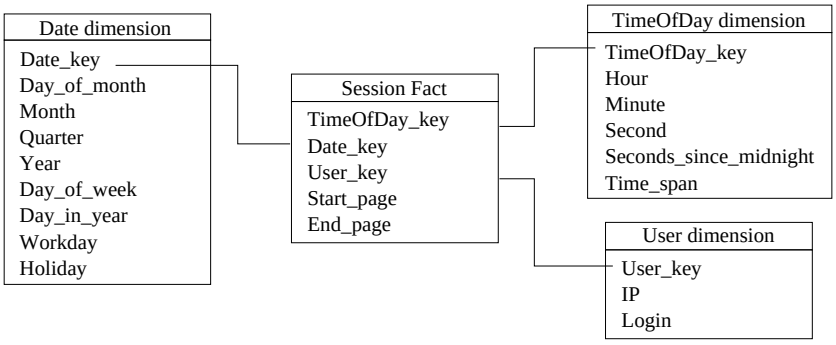

Figure 9: Session fact star scheme

This session fact table has two additional dimensions. The *Start page dimension* captures the first page visited in the session and the *End page dimension* captures the page from which the user leaves the site. Both of these point to a URL dimension table as described in the previous section, but these dimension tables have been left out of the figure for simplicity.

We will not go into further detail about choosing which aggregates to include and which to leave out, as the session fact table is not useable in our case. The advantage of using session facts instead of click facts is that it is easier to follow a user through the site. However, it is impossible to pick out a single click, because this information is lost in the transformation process.

B SQL Queries

This section lists the SQL queries for analyzing subsessions against click and subsession fact tables. The SQL query for finding the most common subsessions of lenghts 2, 3, 4, and 5 against a click fact table, ordered with the most frequently occuring and the longest subsessions first is seen on the next page. The || operator denotes concatenation.

This query is one page of very complicated SQL, which is very hard to write correctly for all but the most seasoned SQL veterans. more importantly, the query has very bad query performance for several reasons. First, the (huge) fact table has to be joined to itself up to five times, as well as to five (smaller) dimension tables. This requires a lot of server resources and because of the complexity of the query, it cannot be efficiently supported by the facilities for pre-computed data (materialized views) found in current DBMS systems. Second, in the sequence join conditions (for instance, c2.number_in_session=c3.number_in_session-1), the "-1" part of the condition means that performance-enhancing techniques, such as using indices for the join process, will not be applicable, leading to very long query processing times. If we wanted to see subsessions of length 2 to 8 instead, the query would be twice as long, with even worse performance.

In contrast, the simple query seen below computes the same answer using a subsession fact table. It only joins the fact table with one (small) dimension table, and the query can be efficiently supported by using pre-computed aggregates.

```
SELECT  us.url_sequence AS start_url,u2.url_name as end_url,
        s.length, COUNT(*) AS occurences
FROM subsession_fact s,url_sequence us
WHERE us.subsession_key=s.subsession_key AND length<=5
```

```
SELECT  url_seqeunce,length,occurences FROM
(
        (SELECT u1.url_name || u2.url_name as url_sequence,
                2 AS length, COUNT(*) AS occurences
        FROM url_dimension u1,url_dimension u2,click_fact c1,click_fact c2
        WHERE c1.number_in_session=c2.number_in_session-1 AND
                c1.session_key = c2.session_key AND
                c1.url_key=u1.url_key AND c2.url_key=u2.url_key
        GROUP BY url_sequence,length)
UNION ALL
        (SELECT u1.url_name||u2.url_name|| u3.url_name as url_sequence,
                3 AS length, COUNT(*) AS occurences
        FROM url_dimension u1,url_dimension u2,url_dimension u3,
                click_fact c1,click_fact c2,click_fact c3
        WHERE c1.number_in_session=c2.number_in_session-1 AND
                c2.number_in_session=c3.number_in_session-1 AND
                c1.session_key = c2.session_key AND c2.session_key = c3.session_key AND
                c1.url_key=u1.url_key AND c2.url_key=u2.url_key AND AND c3.url_key=u3.url_key
        GROUP BY url_sequence,length)
UNION ALL
        (SELECT u1.url_name || u2.url_name || u3.url_name || u4.url_name AS url_sequence,
                4 AS length, COUNT(*) AS occurences
        FROM url_dimension u1,url_dimension u2,url_dimension u3,url_dimension u4,
                click_fact c1,click_fact c2,click_fact c3,click_fact c4
        WHERE c1.number_in_session=c2.number_in_session-1 AND
                c2.number_in_session=c3.number_in_session-1 AND
                c3.number_in_session=c4.number_in_session-1 AND
                c1.session_key = c2.session_key AND c2.session_key = c3.session_key AND
                c3.session_key = c4.session_key
                c1.url_key=u1.url_key AND c2.url_key=u2.url_key) AND
                c3.url_key=u3.url_key AND c4.url_key=u4.url_key
        GROUP BY url_sequence,length)
UNION ALL
        (SELECT u1.url_name || u2.url_name || u3.url_name || u4.url_name || u5.url_name AS url_sequence,
                5 AS length, COUNT(*) AS occurences
        FROM url_dimension u1,url_dimension u2,url_dimension u3,url_dimension u4,url_dimension u5,
                click_fact c1,click_fact, c2,click_fact c3,click_fact c4,click_fact c5
        WHERE c1.number_in_session=c2.number_in_session-1 AND
                c2.number_in_session=c3.number_in_session-1 AND
                c3.number_in_session=c4.number_in_session-1 AND
                c4.number_in_session=c5.number_in_session-1 AND
                c1.session_key = c2.session_key AND c2.session_key = c3.session_key AND
                c3.session_key = c4.session_key AND c4.session_key = c5.session_key
                c1.url_key=u1.url_key AND c2.url_key=u2.url_key) AND
                c3.url_key=u3.url_key AND c4.url_key=u4.url_key) AND
                c5.url_key=u5.url_key
        GROUP BY url_sequence,length)
)
ORDER BY occurences DESC,length DESC,url_sequence ASC
```

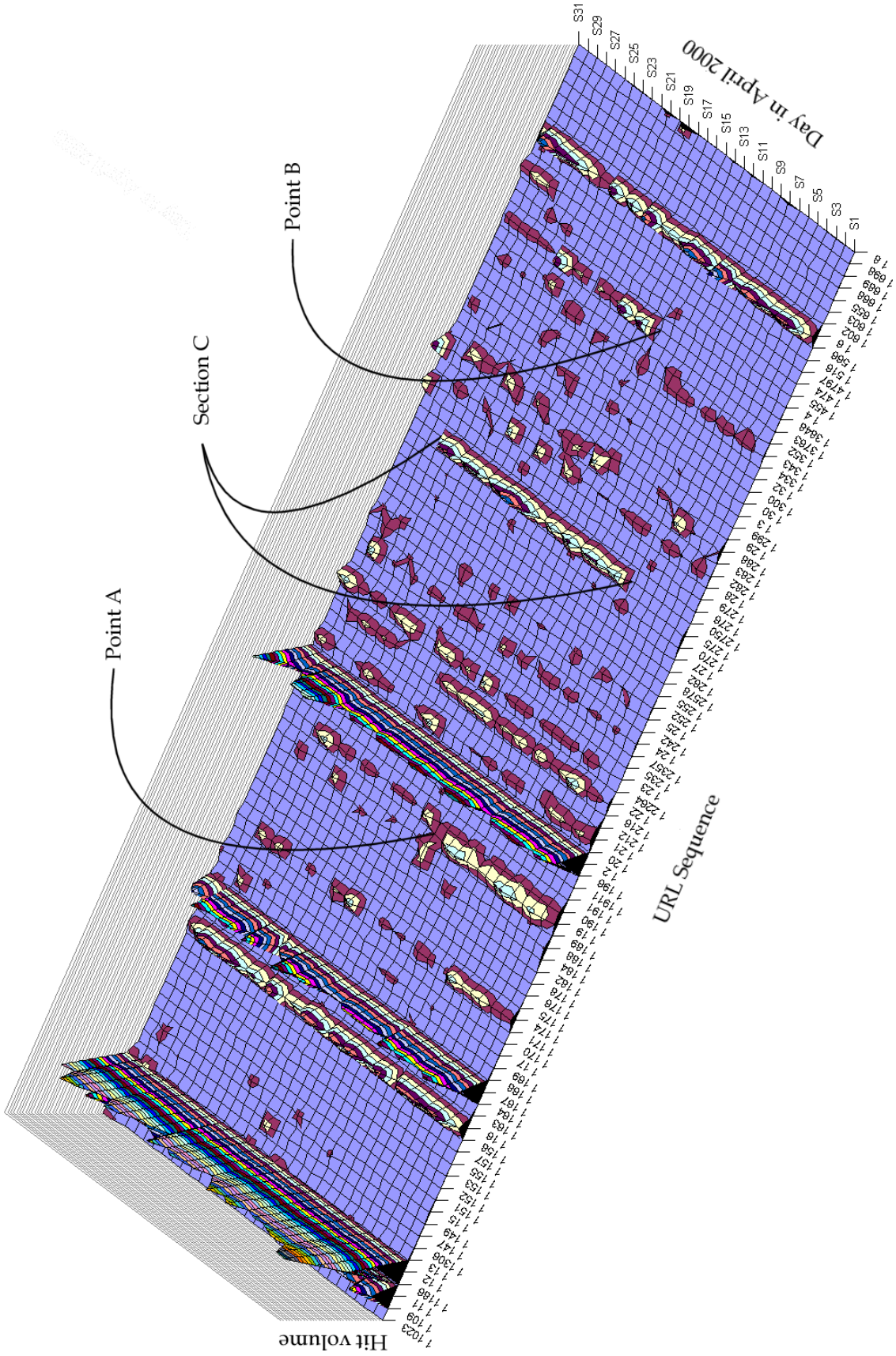

Figure 10: Front Page Banner Target Pages